

DEEP LEARNING COM TENSORFLOW




DEEP FEED FORWARD

DIEGO RODRIGUES DSC

INENET

CRONOGRAMA

Dia	Aula	Trab
02/09	Workshop de Deep Learning	
04/09	Deep FeedForward	
09/09	Rede Neural Convolucional	Modelo Baseline
11/09	AutoEncoder	
16/09	Representation & Transfer Learning	Modelo Profundo
18/09	Sequências	
23/09	Modelos Generativos	Deployment
25/09	Apresentação dos Trabalhos Parte II	

WORKSHOP DE DEEP LEARNING

- PARTE 1 : TEORIA

- DATA PREPARATION

- AUMENTO DE DADOS

- MODELING

- CAPACIDADE
 - CAMADA DE SAÍDA
 - CAMADA OCULTA
 - REGULARIZAÇÃO
 - TREINAMENTO

- PARTE 2 : PRÁTICA

- AMBIENTE
 - LIVRO
 - AVALIAÇÃO
 - TOPOLOGIAS

- PARTE 3 : TRABALHOS

- SEMANA 1

The slide features a light gray gradient background. In the top-left and bottom-right corners, there are clusters of realistic, three-dimensional water droplets of various sizes, some overlapping. A faint, circular watermark is visible in the upper center of the slide.

PARTE 1 : TEORIA

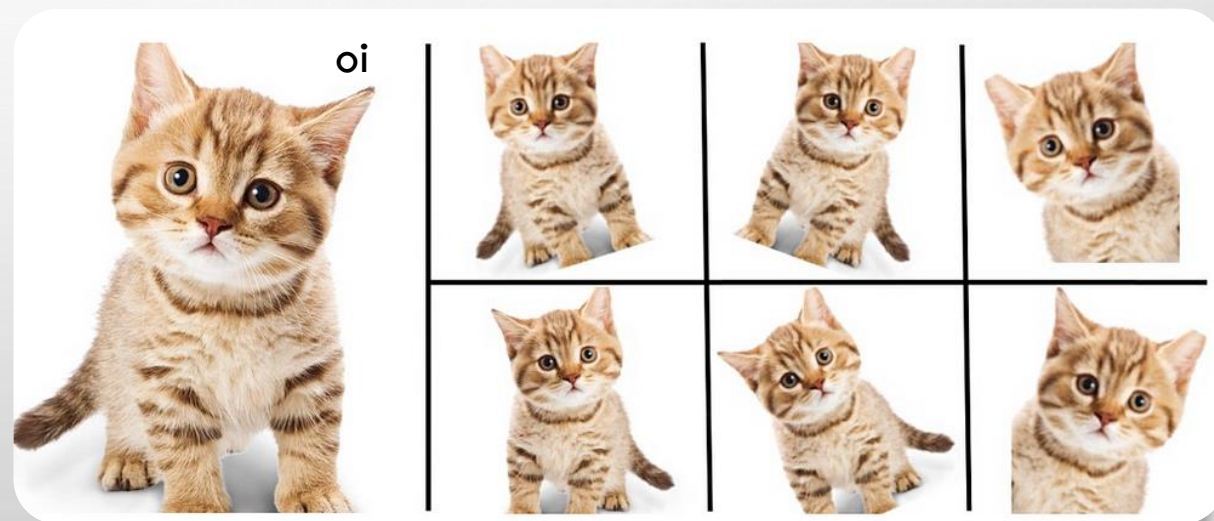
The background is a light gray gradient. In the top-left and bottom-right corners, there are clusters of realistic water droplets of various sizes, rendered with soft shadows and highlights to give them a three-dimensional appearance. In the center of the slide, the text "DATA PREPARATION" is displayed in a bold, black, sans-serif font.

DATA PREPARATION

AUMENTO DE DADOS

Aumento de Dados (Data Augmentation) é uma técnica poderosa utilizada para aumentar a **quantidade e diversidade** do conjunto de dados de treinamento, **sem a necessidade de coletar novos dados**.

Ela é aplicada em vários domínios, como **imagens, texto e áudio**, e envolve a **criação de novas observações** a partir de **re-amostragem ou transformações** nos dados existentes.

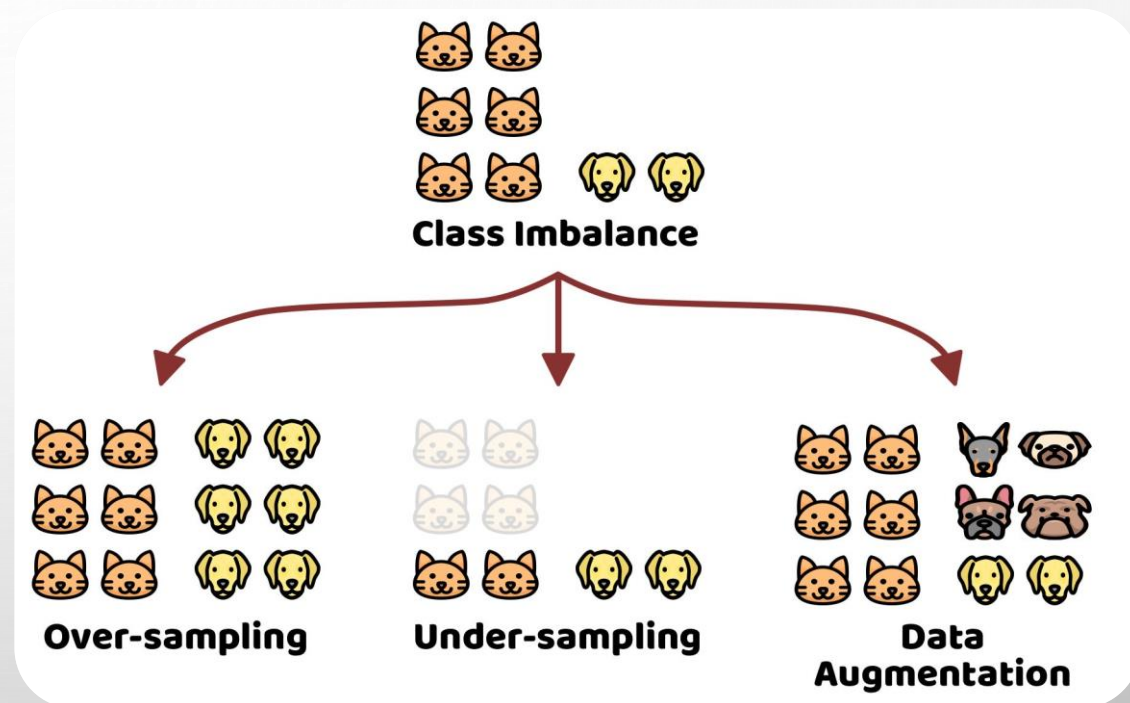


AUMENTO DE DADOS: BALANCEAMENTO DE CLASSES

Aumento de Dados pode ser utilizado para diferentes **estratégias de balanceamento das classes!**

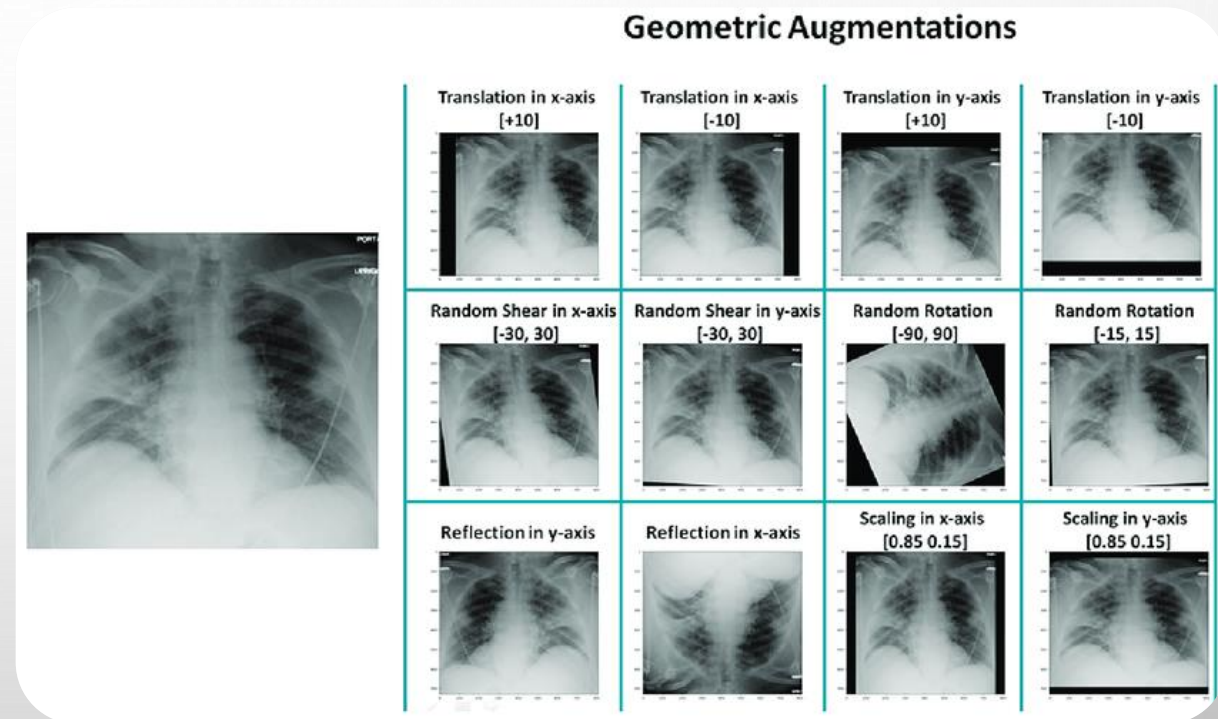
BENEFÍCIOS ADICIONAIS

- **Robustez a Ruído:** Aumentar a capacidade do modelo de lidar com dados ruidosos ou imperfeitos, introduzindo variações que o ajudam a generalizar melhor.
- **Melhoria da Generalização:** Evitar o overfitting ao modelo de treinamento, fornecendo exemplos diversificados que permitem ao modelo aprender características mais robustas.



AUMENTO DE DADOS: TIPOS DE TRANSFORMAÇÕES

- **Transformações Geométricas:** Como rotações, translações, e alterações de escala, aplicadas a diferentes tipos de dados.
- **Alterações em Valores:** Como mudança na intensidade de pixels em imagens, adição de ruído em dados de áudio, ou substituição de palavras em textos.
- **Composição e Mistura:** Combinação de diferentes amostras de dados para criar novos exemplos compostos.



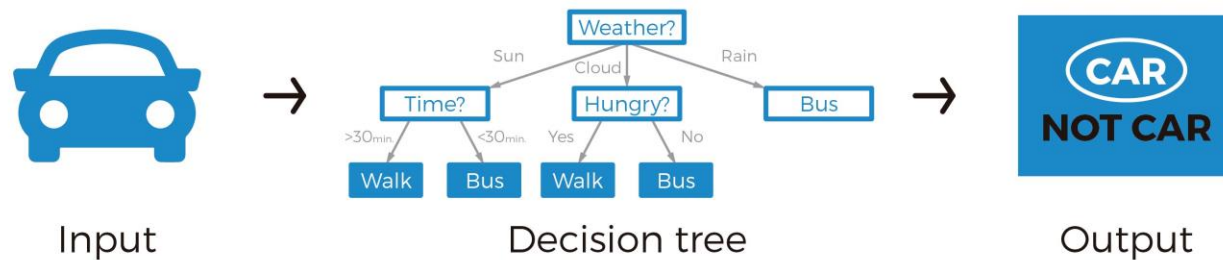
MODELING



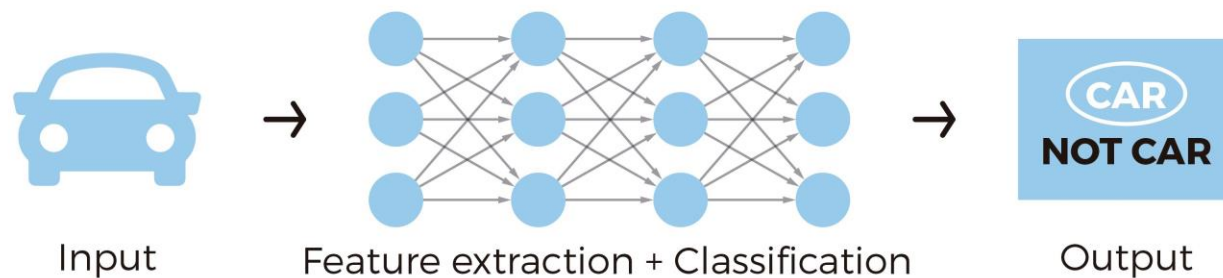
CAPACIDADE

APRENDIZADO PROFUNDO

Machine Learning



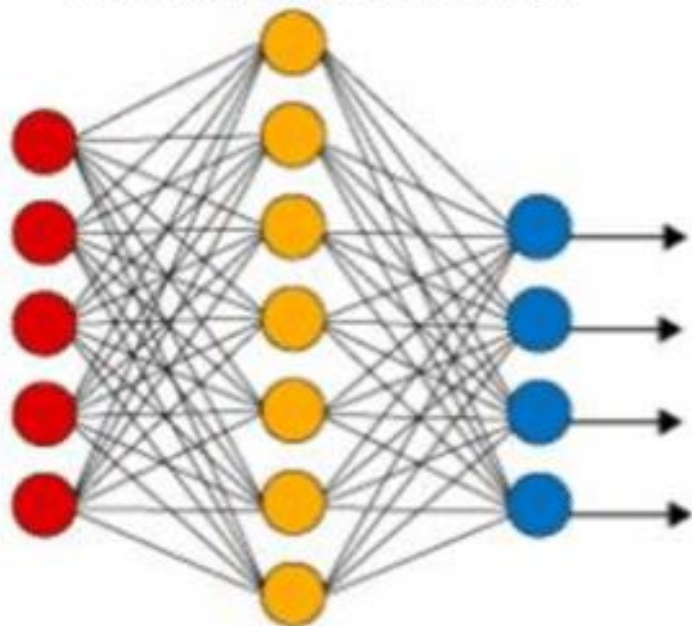
Deep Learning



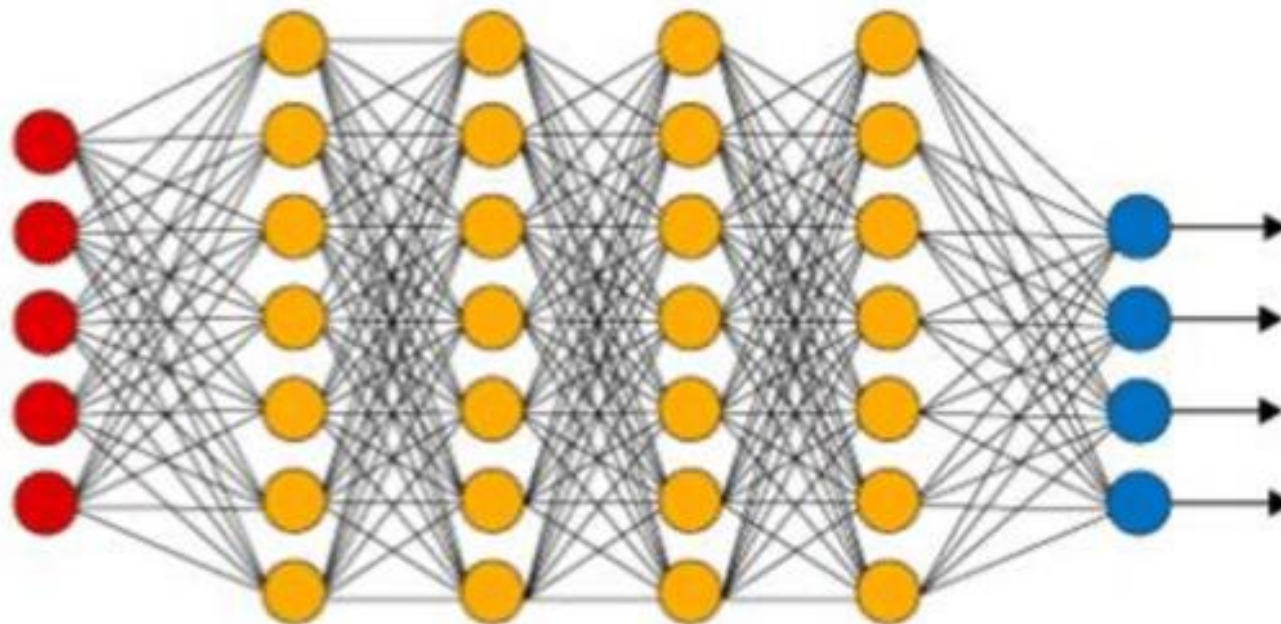
LARGURA E PROFUNDIDADE

Largura

Artificial Neural Network



Deep Neural Network



Profundidade

LARGURA E PROFUNDIDADE

CHAPTER 6. DEEP FEEDFORWARD NETWORKS

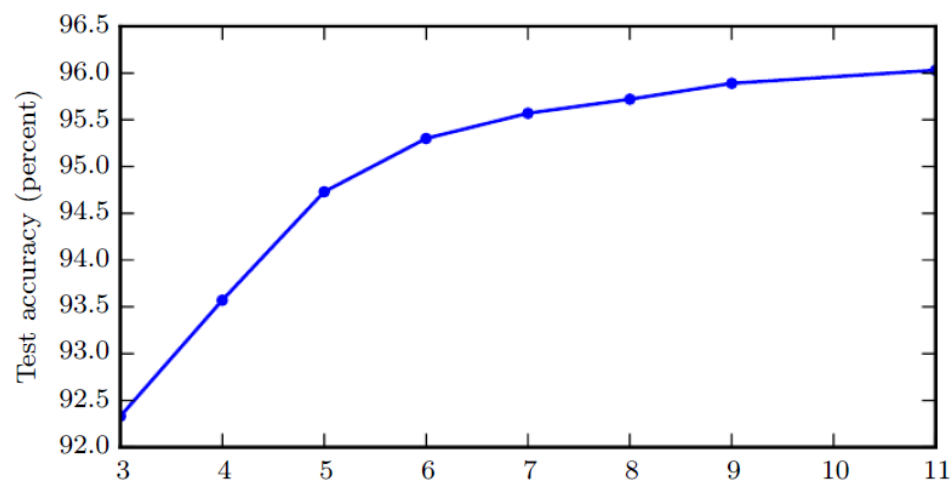


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from [Goodfellow et al. \(2014d\)](#). The test set accuracy consistently increases with increasing depth. See figure 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

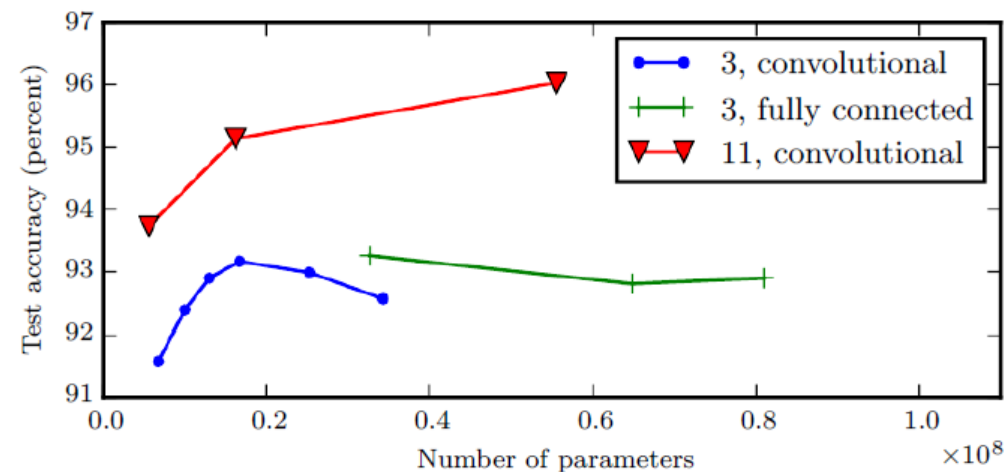


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from [Goodfellow et al. \(2014d\)](#) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler representations (e.g., corners defined in terms of edges) or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other, then recognize them).

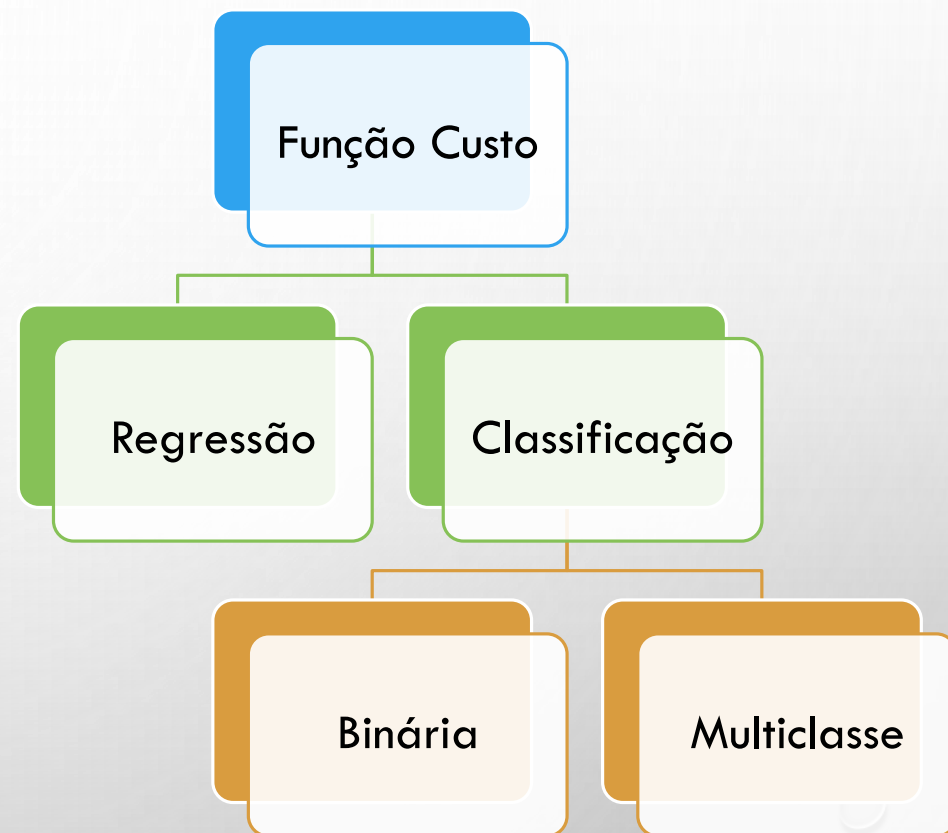


CAMADA DE SAÍDA

FUNÇÕES CUSTO

A **escolha da função custo** (ou função de perda) em um modelo de deep learning é crucial e deve estar **alinhada com a função de ativação da camada de saída e a natureza do problema**.

Dependendo do **tipo de tarefa**, diferentes **combinações de ativação e função custo** são recomendadas para **otimizar o desempenho do modelo**.



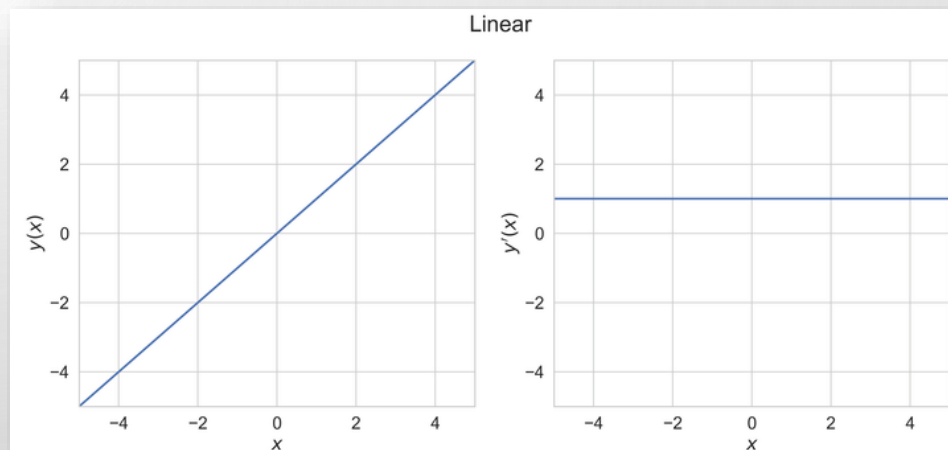
FUNÇÃO CUSTO: REGRESSÃO

Em **Regressão**, onde o objetivo é prever um valor contínuo, a **Ativação Linear** é a escolha mais adequada. **Mean Squared Error (MSE)** é a função custo mais utilizada nesse contexto, pois mede a média dos quadrados dos erros entre os valores preditos pelo modelo e os valores reais, considerando a **Distribuição Gaussiana**.

Ativação Linear: A saída é um valor contínuo que pode assumir qualquer valor real.

MSE (Mean Squared Error)

- Recomendada quando as variáveis seguem uma distribuição gaussiana.
- Penaliza erros grandes de forma mais severa, sendo sensível a outliers.



$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

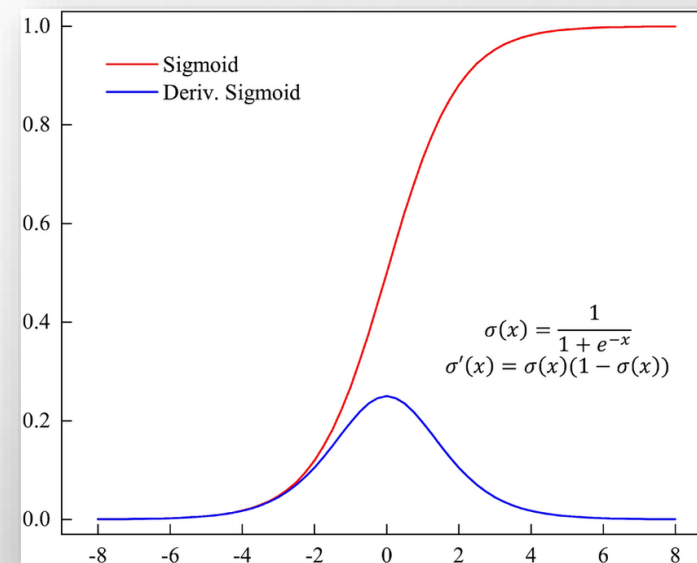
FUNÇÃO CUSTO: CLASSIFICAÇÃO BINÁRIA

Em **Classificação Binária**, onde o objetivo é prever uma das duas possíveis classes, a **ativação sigmoide** é frequentemente utilizada. A função custo associada é a **Entropia Binária Cruzada (Binary Cross-Entropy)**, que mede a divergência entre as probabilidades preditas e as probabilidades reais das classes, considerando a **Distribuição de Bernoulli**.

Ativação Sigmoide: A saída é uma probabilidade entre 0 e 1, representando a chance de pertencer a uma das classes.

Entropia Binária Cruzada

- Baseada no princípio da máxima verossimilhança.
- Adequada para problemas onde as classes são mutuamente exclusivas e a saída é binária.



$$\text{BCE} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

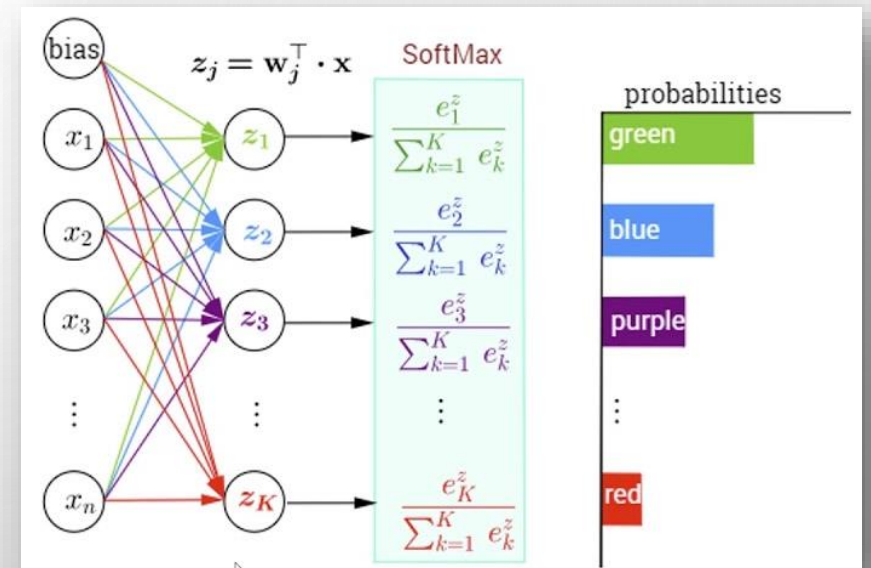
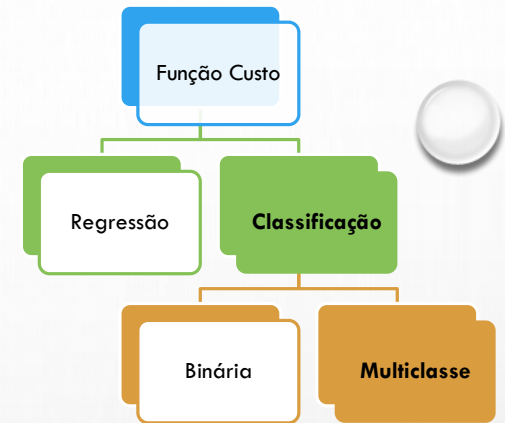
FUNÇÃO CUSTO: CLASSIFICAÇÃO MULTICLASSE

Em **Classificação Multiclasse**, onde o objetivo é prever uma classe entre várias possíveis, a ativação softmax é a mais indicada. A função custo associada é a **Entropia Cruzada Categórica (Categorical Cross-Entropy)**, que avalia a diferença entre a distribuição de probabilidade predita e a verdadeira, considerando a **Distribuição Multinoulli**.

Ativação Softmax: A saída é uma distribuição de probabilidade sobre as classes, somando 1.

Categorical Cross-Entropy

- Também fundamentada no princípio da máxima verossimilhança.
- Ideal para problemas onde cada entrada pertence a uma única classe em um conjunto de classes.



$$CCE = - \sum_{i=1}^n y_i \log(\hat{y}_i)$$



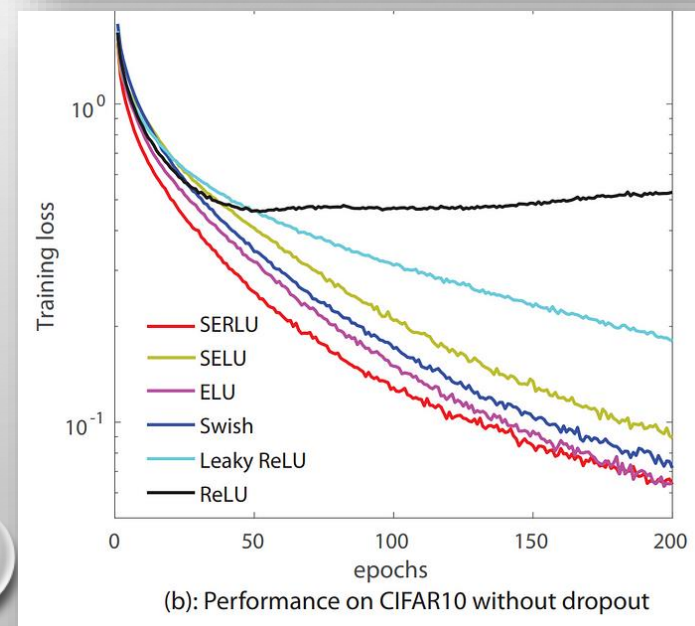
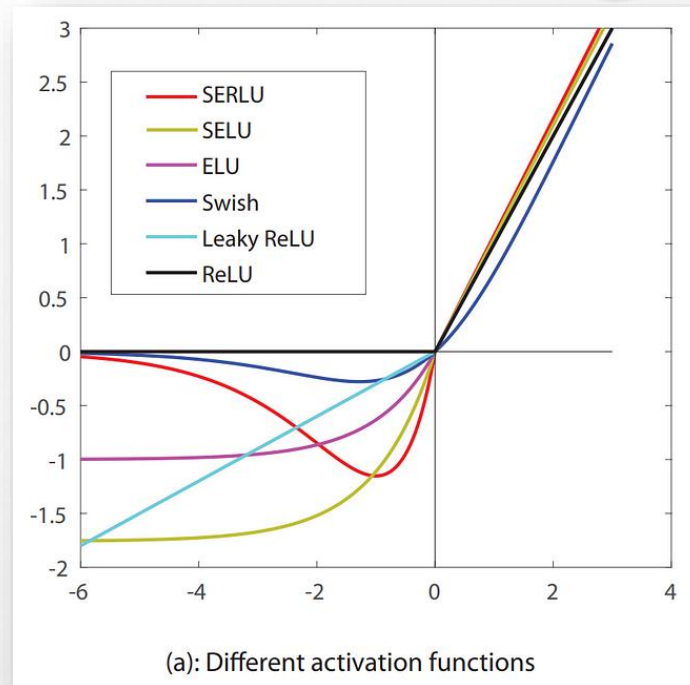
CAMADA OCULTA

FAMÍLIA RELU

A **Família ReLU (Rectified Linear Unit)** é uma das mais populares em redes neurais profundas devido à sua **simplicidade** e **eficiência** no treinamento de modelos complexos.

Características Gerais:

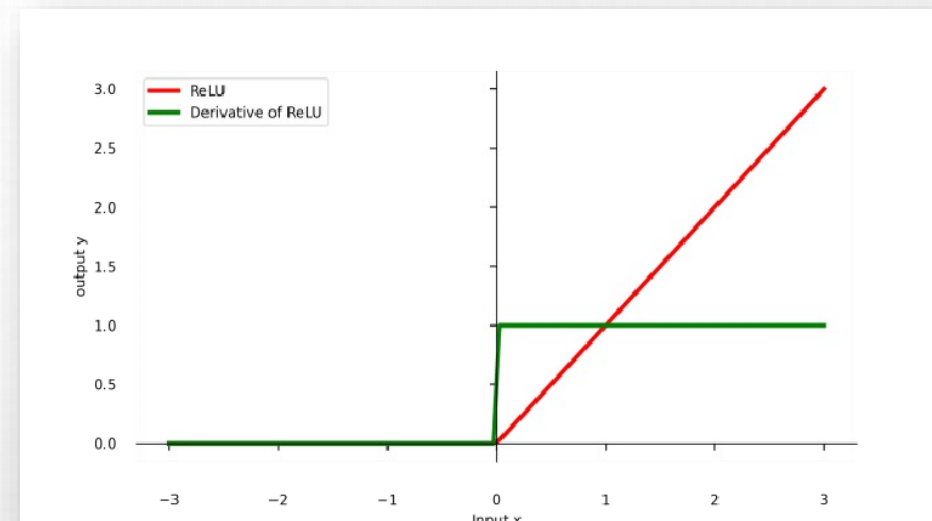
- As ativações ReLU são computacionalmente eficientes.
- Aumentam a esparsidade das ativações, o que pode levar a redes mais interpretáveis e eficientes.
- Ideal para deep learning devido à boa performance em redes profundas.



FAMÍLIA RELU

ReLU (Rectified Linear Unit)

- **Vantagens:** Facilita a convergência durante o treinamento e reduz problemas de dissipação do gradiente.
- **Limitações:** "Morte de neurônios" quando muitos outputs ficam zerados.



$$\text{ReLU}(x) = \max(0, x)$$

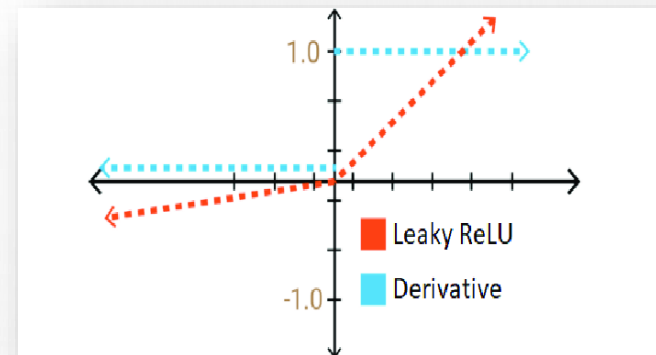
FAMÍLIA RELU

Leaky ReLU:

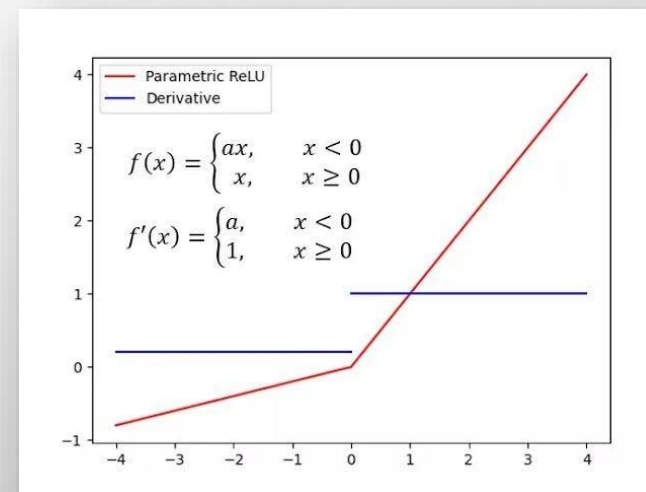
- **Vantagens:** Mitiga o problema da "morte de neurônios" permitindo valores negativos pequenos.

Parametric ReLU (PReLU)

- **Vantagens:** Oferece maior flexibilidade, ajustando a inclinação para valores negativos durante o treinamento.



$$\text{Leaky ReLU}(x) = \max(0.01x, x)$$

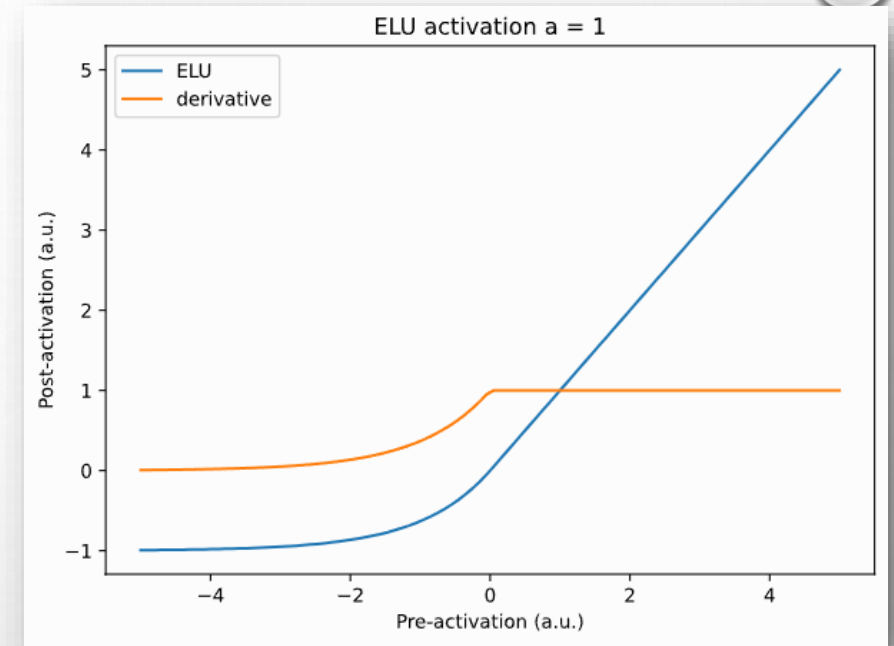


$$\text{PReLU}(x) = \max(\alpha x, x)$$

FAMÍLIA RELU

Exponential Linear Unit (ELU)

- **Vantagens:** Introduz suavidade na ativação, o que pode melhorar a aprendizagem em algumas situações.

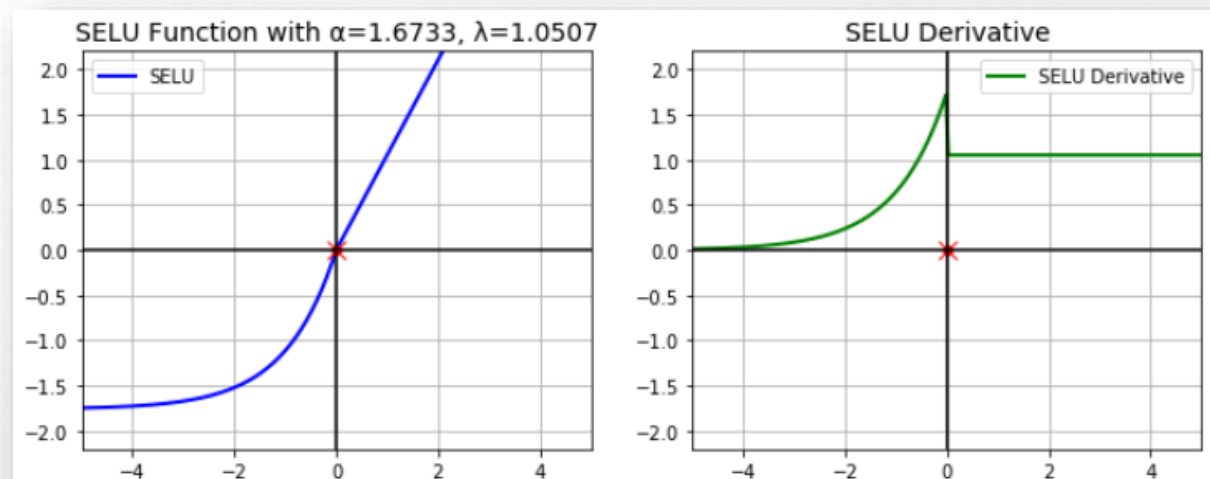


$$ELU(x) = x \text{ se } x > 0; \alpha(\exp(x) - 1) \text{ se } x \leq 0.$$

FAMÍLIA RELU

Scaled Exponential Linear Unit (SeLU)

- **Vantagens:** Automatiza o ajuste de pesos, ajudando a manter a normalização das ativações em camadas profundas, o que pode ser crucial para o auto-escalamento de redes neurais.
- **Usos:** SeLUs são frequentemente usados em redes profundas sem camadas de normalização explícitas.

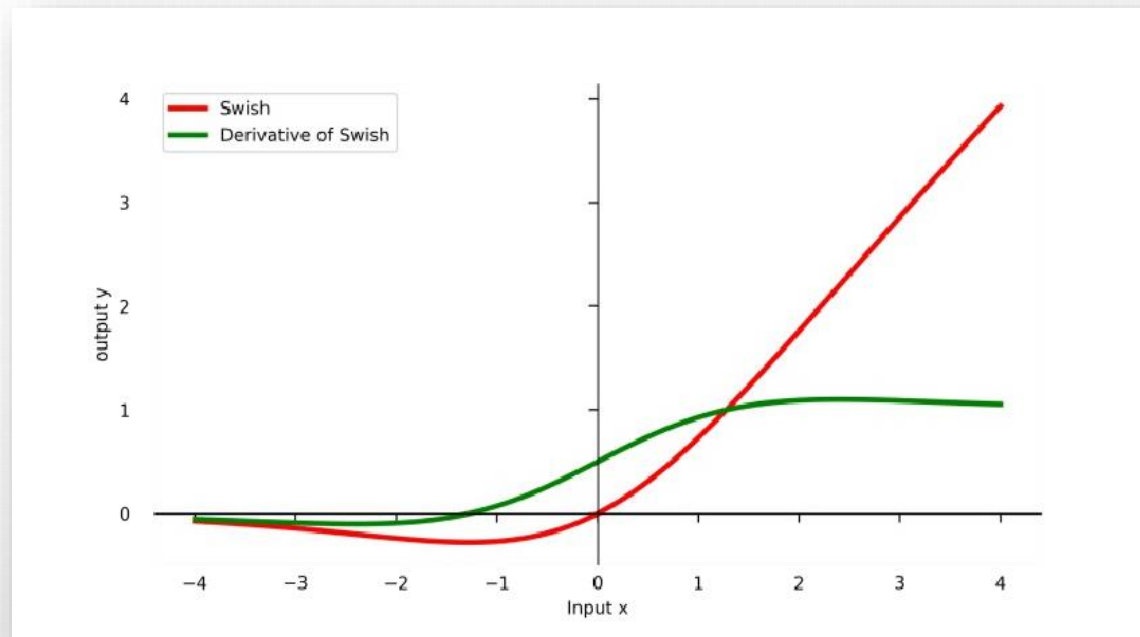


$$\text{SeLU}(x) = \lambda(x) \text{ se } x > 0; \lambda\alpha(\exp(x) - 1) \text{ se } x \leq 0,$$

FAMÍLIA RELU

Swish

- **Vantagens:** Oferece uma ativação suave e contínua que permite gradientes mais fortes para valores negativos, melhorando a performance em algumas arquiteturas de deep learning.
- **Propriedades:** Combina características de ReLU e Sigmoid, com uma leve não-linearidade que pode resultar em melhor desempenho em certos cenários.
- **Desenvolvimento:** Introduzida por pesquisadores do Google, Swish tem mostrado ser eficaz em tarefas de visão computacional.



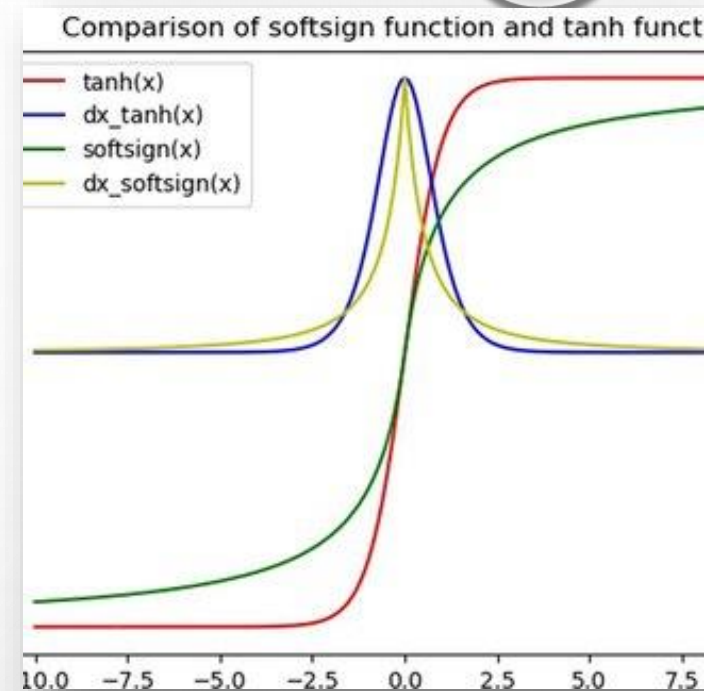
$$\text{Swish}(x) = x \cdot \text{sigmoid}(x) = \frac{x}{1+e^{-x}}$$

FAMÍLIA TANH

A função de ativação **Tangente Hiperbólica (Tanh)** e a **Softsign** pertencem a um grupo de funções de ativação que são **simétricas em torno de zero**, o que significa que elas mapeiam entradas negativas para saídas negativas e entradas positivas para saídas positivas. Isso pode **ajudar no treinamento de redes neurais**, especialmente em camadas ocultas, ao **centralizar as ativações em torno de zero**.

Características Gerais

- **Centralização em Zero:** Ambas as funções têm saídas centradas em torno de zero, o que pode acelerar o treinamento ao facilitar a propagação do gradiente em redes neurais.
- **Suavidade:** Essas ativações produzem transições suaves entre -1 e 1, o que pode ajudar a evitar saltos abruptos nas ativações.
- **Redução do Problema de Saturação:** Embora possam sofrer com a dissipação do gradiente em valores extremos, suas saídas simétricas e centralizadas oferecem gradientes mais robustos em comparação à sigmoide.
- **Aplicação em Redes Recorrentes:** Devido à sua centralização e suavidade, são preferidas em redes recorrentes (RNNs) e em situações onde a saída contínua entre -1 e 1 é desejável.
- **Computacionalmente Mais Pesadas:** Comparadas a funções como ReLU, Tanh e Softsign são mais computacionalmente exigentes, mas oferecem benefícios que podem justificar seu uso em certos cenários.



FAMÍLIA TANH

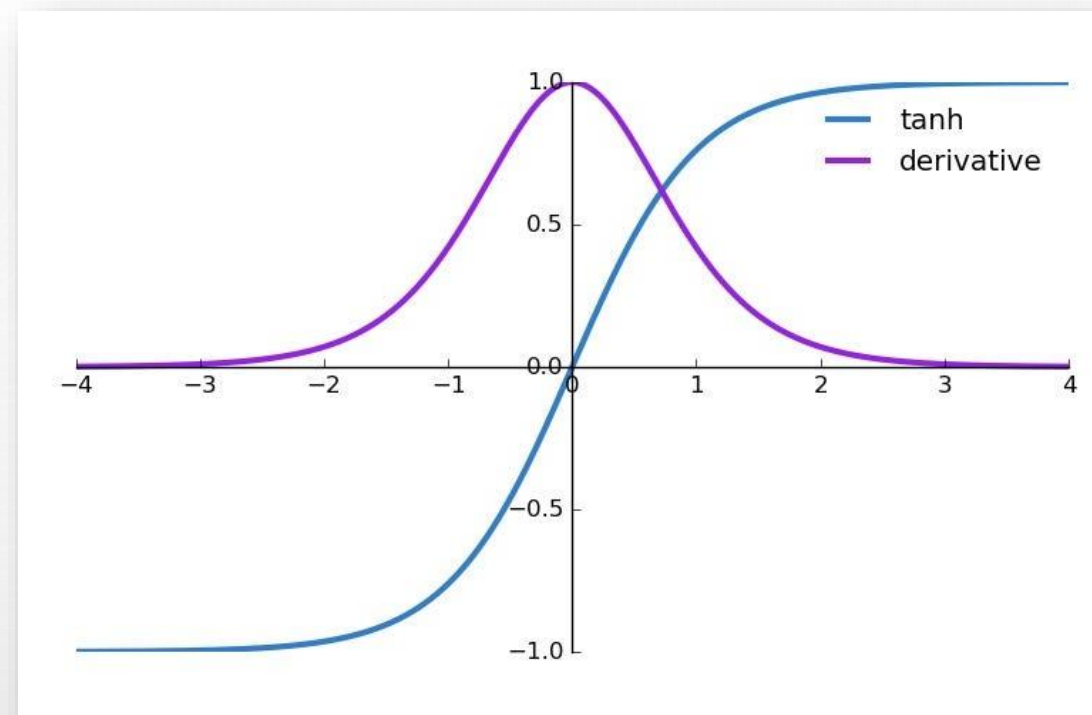
Tangente Hiperbólica (Tanh)

- **Vantagens**

- Centraliza os dados em torno de zero, facilitando o treinamento.
- Fornece gradientes mais fortes em valores extremos comparados à função sigmoide.
- Usada frequentemente em redes recorrentes (RNNs) e camadas ocultas em redes feedforward.

- **Limitações**

- Pode sofrer do problema de desvanecimento do gradiente em valores extremos.
- Mais computacionalmente custosa que ReLU.

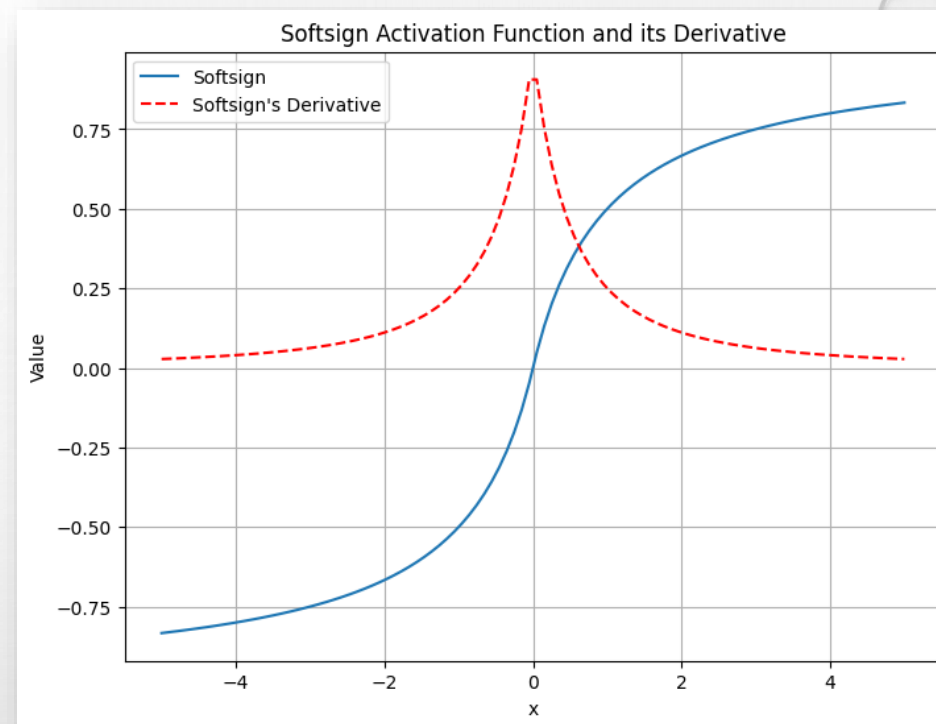


$$\text{Tanh}(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

FAMÍLIA TANH

Softsign

- **Vantagens**
 - Semelhante ao Tanh, mas com uma curva mais suave que aproxima o zero de forma mais lenta.
 - Pode ser uma alternativa menos agressiva ao Tanh em certos cenários, especialmente onde a suavidade adicional pode beneficiar o treinamento.
 - Centraliza as ativações em torno de zero.
- **Limitações**
 - Embora menos propensa a desvanecer gradientes do que Tanh, ainda é suscetível em entradas muito grandes ou muito pequenas.



$$\text{Softsign}(x) = \frac{x}{1+|x|}$$



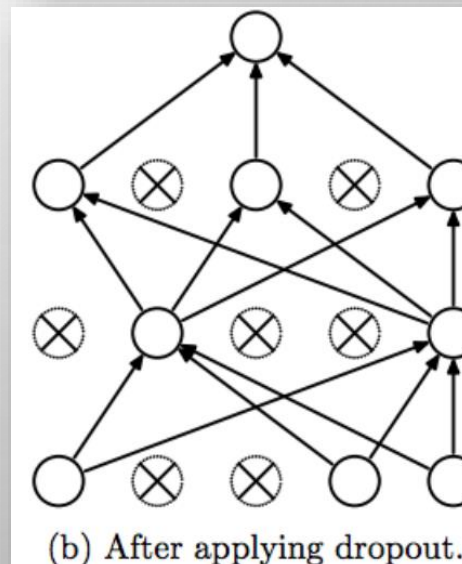
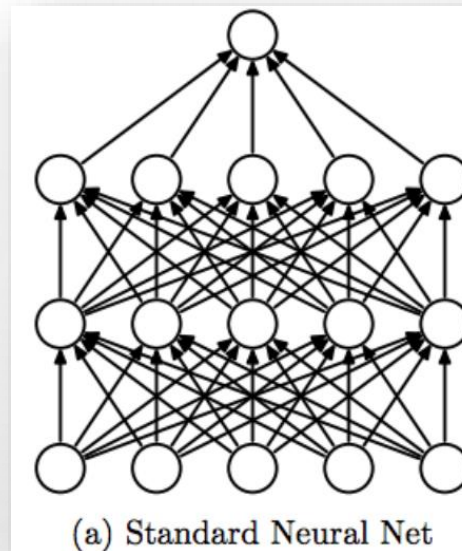
REGULARIZAÇÃO

DROPOUT

É uma **técnica de regularização** usada para **prevenir o overfitting** em redes neurais profundas. Durante o treinamento, o Dropout **desativa aleatoriamente uma fração de neurônios em cada camada**, forçando a rede a **não depender de neurônios específicos** para o aprendizado. Isso incentiva a rede a aprender **representações mais robustas e generalizáveis**.

Como funciona

- Durante cada iteração de treinamento, **uma proporção p dos neurônios em uma camada é desativada**.



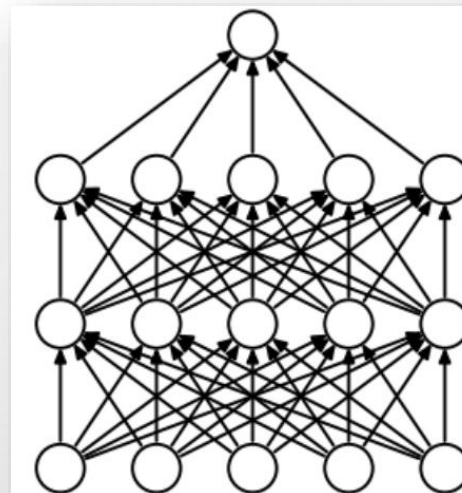
DROPOUT

Vantagens

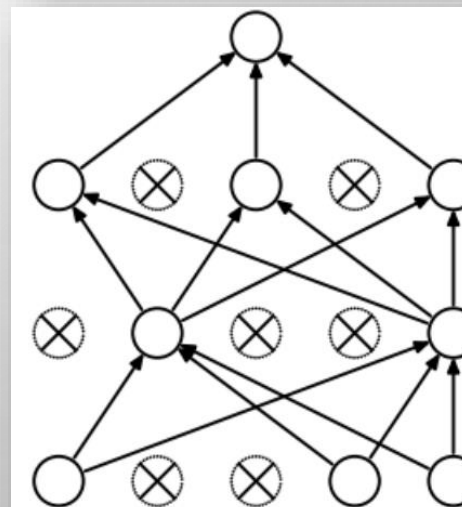
- Reduz o overfitting ao impedir que a rede se torne excessivamente dependente de neurônios específicos.
- Simples de implementar e eficaz em diversas arquiteturas.

Limitações

- Pode aumentar o tempo de treinamento devido à necessidade de mais iterações para convergência.
- Não é necessário em modelos que já utilizam outras formas de regularização intensa, como redes que fazem uso de Batch Normalization.



(a) Standard Neural Net



(b) After applying dropout.

DROPOUT

Dropout layer

Dropout class

[\[source\]](#)

```
keras.layers.Dropout(rate, noise_shape=None, seed=None, **kwargs)
```

Applies dropout to the input.

The `Dropout` layer randomly sets input units to 0 with a frequency of `rate` at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by $1 / (1 - \text{rate})$ such that the sum over all inputs is unchanged.

Note that the `Dropout` layer only applies when `training` is set to `True` in `call()`, such that no values are dropped during inference. When using `model.fit`, `training` will be appropriately set to `True` automatically. In other contexts, you can set the argument explicitly to `True` when calling the layer.

(This is in contrast to setting `trainable=False` for a `Dropout` layer. `trainable` does not affect the layer's behavior, as `Dropout` does not have any variables/weights that can be frozen during training.)

Arguments

- **rate**: Float between 0 and 1. Fraction of the input units to drop.
- **noise_shape**: 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape `(batch_size, timesteps, features)` and you want the dropout mask to be the same for all timesteps, you can use `noise_shape=(batch_size, 1, features)`.
- **seed**: A Python integer to use as random seed.

Call arguments

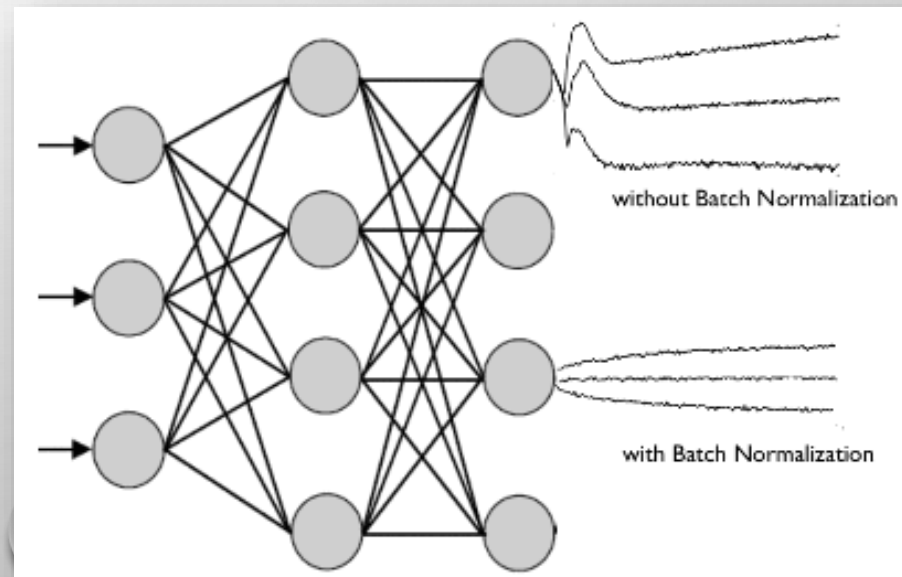
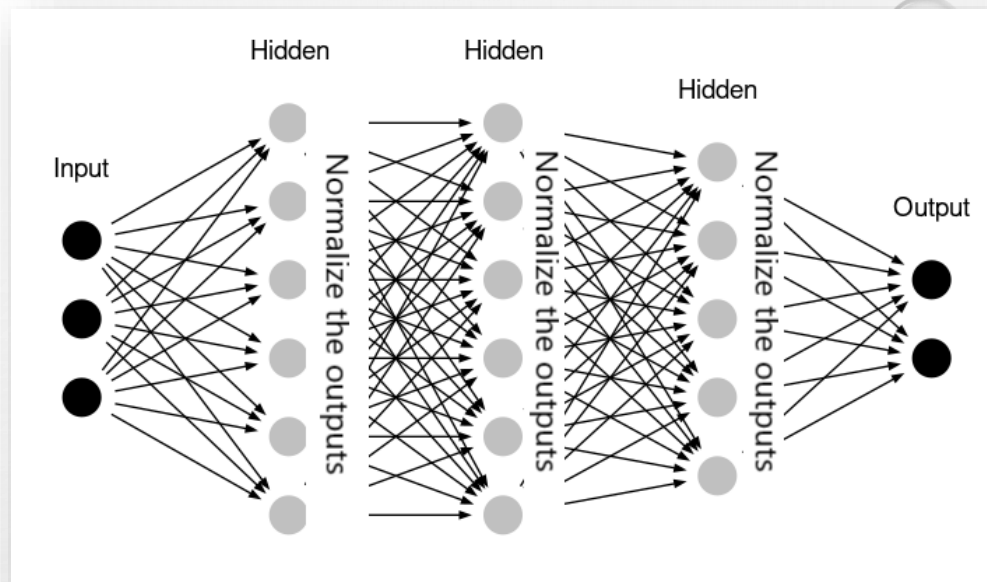
- **inputs**: Input tensor (of any rank).
- **training**: Python boolean indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

BATCH NORMALIZATION

É uma técnica que normaliza as ativações de cada camada em mini-batches durante o treinamento, o que acelera o aprendizado e melhora a estabilidade do modelo. Ela atua reduzindo a covariância do deslocamento (shift) das ativações, o que permite usar taxas de aprendizado maiores e menos sensibilidade à inicialização dos pesos.

Como funciona

- Normaliza as ativações de uma camada para que tenham média zero e variância unitária dentro de cada mini-batch.
- Inclui parâmetros aprendíveis (escala e deslocamento) para restaurar a representatividade original das ativações, se necessário.



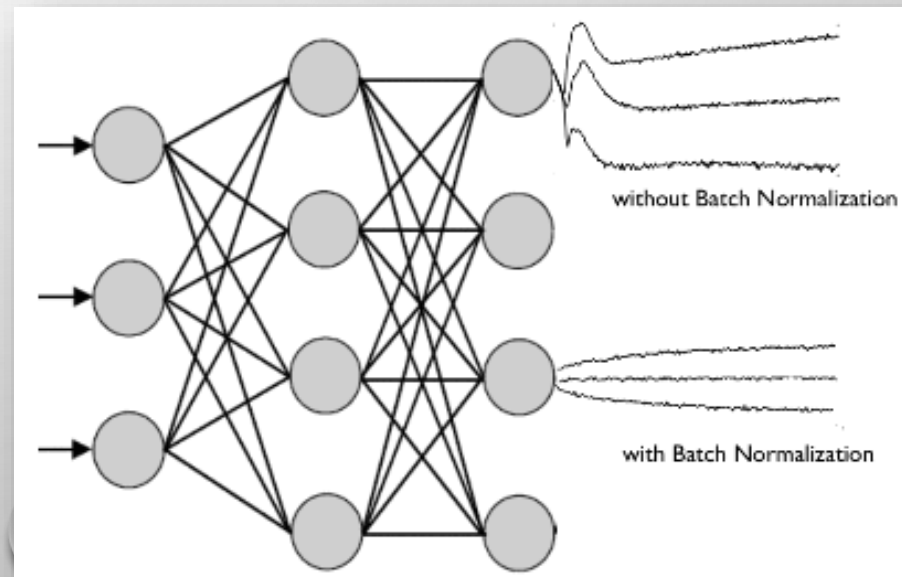
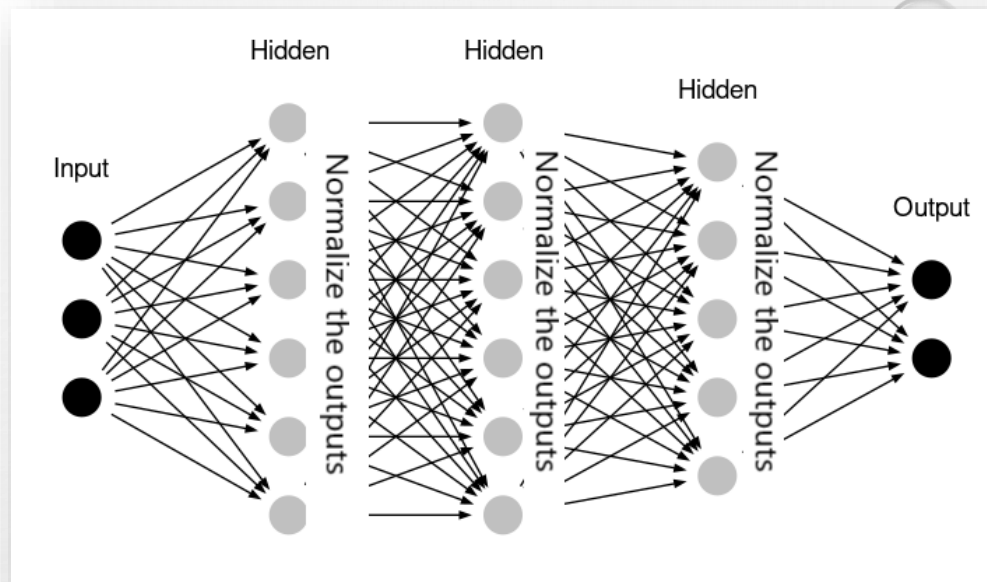
BATCH NORMALIZATION

Vantagens

- Acelera o treinamento ao permitir o uso de taxas de aprendizado maiores.
- Reduz a sensibilidade à inicialização dos pesos e ao ajuste de hiperparâmetros.
- Pode ter efeito de regularização, reduzindo o overfitting.

Limitações

- Introduz complexidade computacional adicional durante o treinamento.
- Pode ser menos eficaz em problemas onde a normalização dos dados de entrada já é adequada ou onde o Dropout é mais apropriado.



BATCH NORMALIZATION

BatchNormalization layer

BatchNormalization class

[\[source\]](#)

```
keras.layers.BatchNormalization(  
    axis=-1,  
    momentum=0.99,  
    epsilon=0.001,  
    center=True,  
    scale=True,  
    beta_initializer="zeros",  
    gamma_initializer="ones",  
    moving_mean_initializer="zeros",  
    moving_variance_initializer="ones",  
    beta_regularizer=None,  
    gamma_regularizer=None,  
    beta_constraint=None,  
    gamma_constraint=None,  
    synchronized=False,  
    **kwargs  
)
```

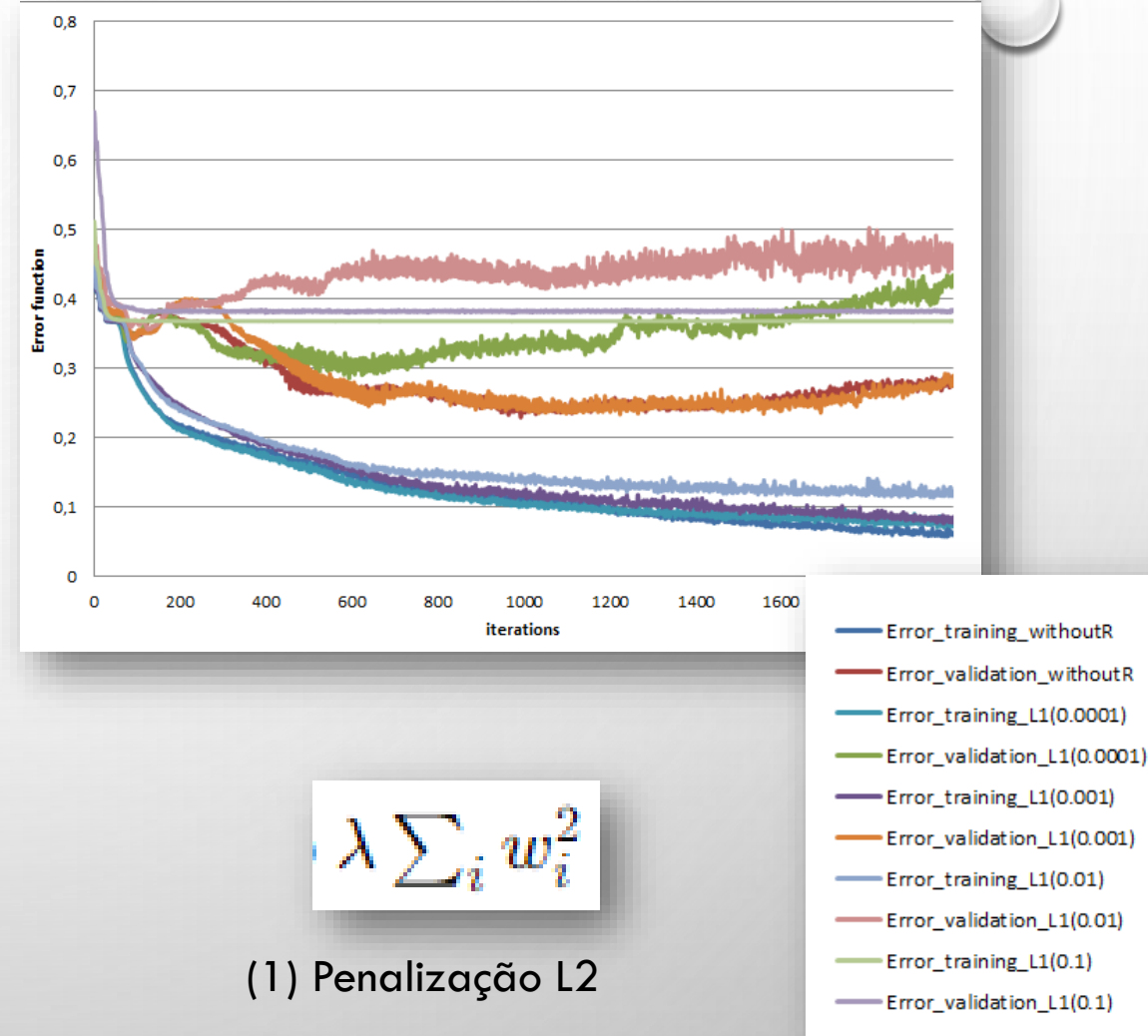
Layer that normalizes its inputs.

Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

NORMA L2

Norma L2 (Regularização Ridge)

- **Definição:** Adiciona um termo de penalização (1) à função de perda, onde λ é o coeficiente de regularização e w_i são os pesos da rede.
- **Efeito:** Penaliza pesos grandes, incentivando uma distribuição mais uniforme dos pesos e ajudando a prevenir o overfitting.
- **Vantagens:** Mantém pesos pequenos e distribuídos, resultando em modelos menos complexos e mais generalizáveis.
- **Aplicação:** Usada quando é desejável que todos os pesos tenham alguma influência, mas em magnitudes menores.



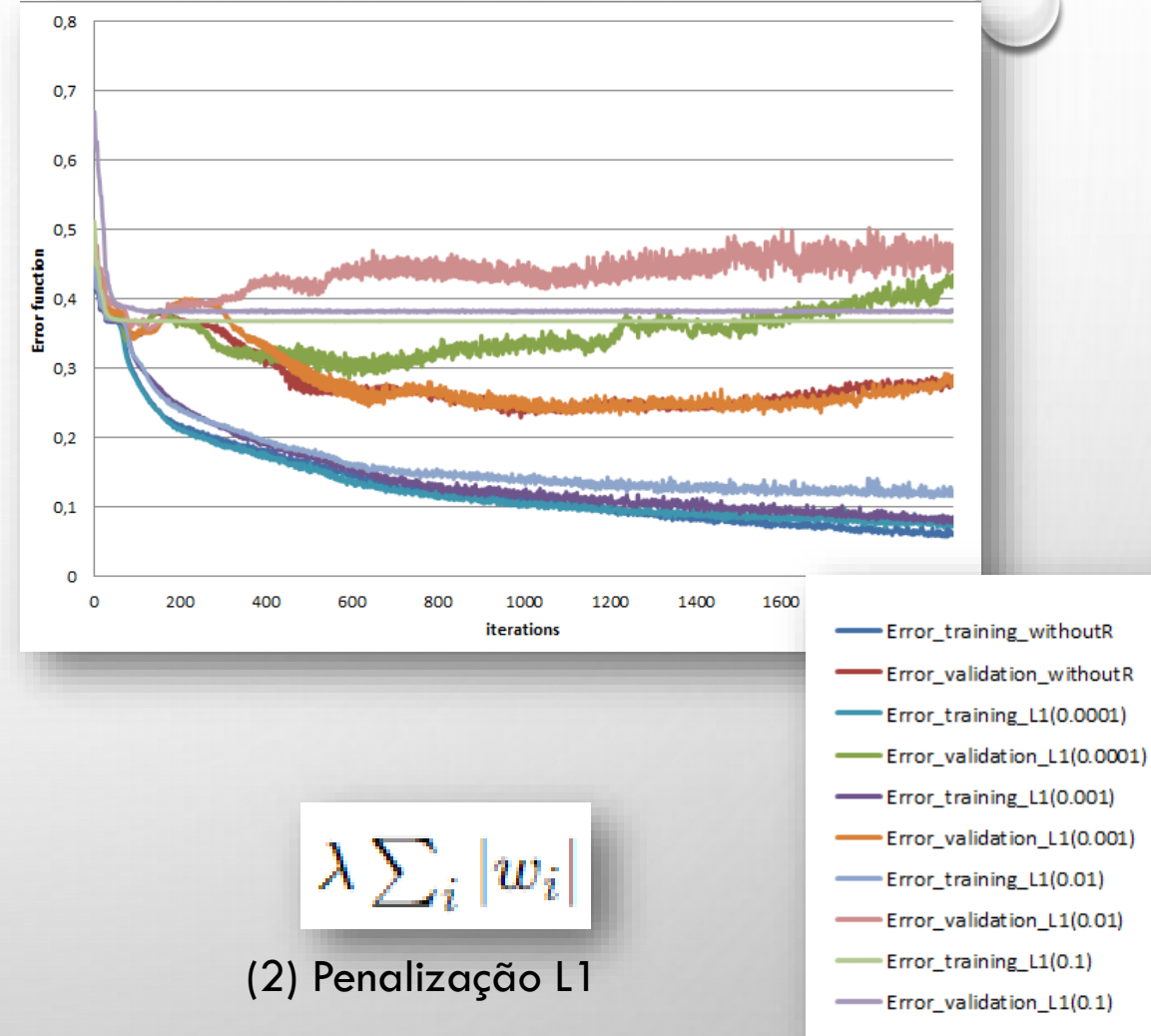
$$\lambda \sum_i w_i^2$$

(1) Penalização L2

NORMA L1

Norma L1 (Regularização Lasso)

- **Definição:** Adiciona um termo (2) à função de perda, penalizando a soma dos valores absolutos dos pesos.
- **Efeito:** Promove esparsidade nos pesos, incentivando que muitos pesos se tornem exatamente zero, o que pode levar à seleção automática de atributos.
- **Vantagens:** Cria modelos mais simples e interpretáveis, útil para seleção de atributos.
- **Aplicação:** Preferida em cenários onde a esparsidade é desejada, como em problemas de alta dimensionalidade com muitos atributos irrelevantes.



$$\lambda \sum_i |w_i|$$

NORMA L1 E L2

Layer weight regularizers

Regularizers allow you to apply penalties on layer parameters or layer activity during optimization. These penalties are summed into the loss function that the network optimizes.

Regularization penalties are applied on a per-layer basis. The exact API will depend on the layer, but many layers (e.g. `Dense`, `Conv1D`, `Conv2D` and `Conv3D`) have a unified API.

These layers expose 3 keyword arguments:

- `kernel_regularizer`: Regularizer to apply a penalty on the layer's kernel
- `bias_regularizer`: Regularizer to apply a penalty on the layer's bias
- `activity_regularizer`: Regularizer to apply a penalty on the layer's output

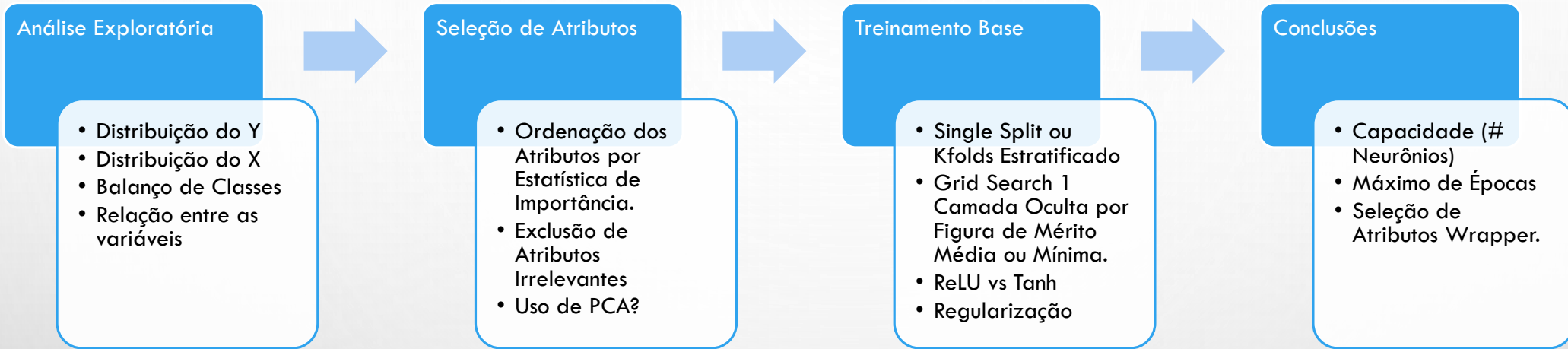
```
from keras import layers
from keras import regularizers

layer = layers.Dense(
    units=64,
    kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
    bias_regularizer=regularizers.L2(1e-4),
    activity_regularizer=regularizers.L2(1e-5)
)
```




TREINAMENTO

HEURÍSTICA DE TREINAMENTO



Algorithm 7.2 A meta-algorithm for using early stopping to determine how long to train, then retraining on all the data.

Let $\mathbf{X}^{(\text{train})}$ and $\mathbf{y}^{(\text{train})}$ be the training set.

Split $\mathbf{X}^{(\text{train})}$ and $\mathbf{y}^{(\text{train})}$ into $(\mathbf{X}^{(\text{subtrain})}, \mathbf{X}^{(\text{valid})})$ and $(\mathbf{y}^{(\text{subtrain})}, \mathbf{y}^{(\text{valid})})$ respectively.

Run early stopping (algorithm 7.1) starting from random θ using $\mathbf{X}^{(\text{subtrain})}$ and $\mathbf{y}^{(\text{subtrain})}$ for training data and $\mathbf{X}^{(\text{valid})}$ and $\mathbf{y}^{(\text{valid})}$ for validation data. This returns i^* , the optimal number of steps.

Set θ to random values again.

Train on $\mathbf{X}^{(\text{train})}$ and $\mathbf{y}^{(\text{train})}$ for i^* steps.



The background is a light gray gradient. In the top-left and bottom-right corners, there are several realistic water droplets of various sizes, rendered with soft shadows and highlights to give them a three-dimensional appearance. In the center of the page, there is a faint, circular watermark. It contains a stylized graphic of a person with arms raised, surrounded by text that is difficult to read but appears to be in Portuguese.

PARTE 2 : PRÁTICA

AMBIENTE PYTHON



6. Machine Learning

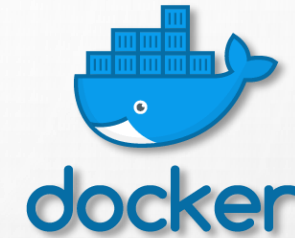


6. Deployment



4. Variáveis Aleatórias

5. Visualização



1. Editor de Código



2. Gestor de Ambiente



3. Ambiente Python do Projeto



3. Notebook Dinâmico

WORKSHOP

- **QUAL A TOPOLOGIA DE DEEP LEARNING ADEQUADA PARA O MEU TRABALHO?**
- **QUAL CAPÍTULO DO LIVRO MELHOR SE ENQUADRA NO MEU TRABALHO?**
- **AULA 3: NOVO CICLO DE BUSINESS UNDERSTANDING / GRUPO + MODELO BASELINE TREINADO**
- AULA 5 OU 7: MODELO PROFUNDO TREINADO
- AULA 7: DEPLOYMENT DO MODELO*
- AULA 3-7 > APRESENTAÇÃO TEÓRICA DA(S) TOPOLOGIA(S) + LEITURA DE ARTIGO + ACOMPANHAMENTO DOS TRABALHOS +DEEP DIVE NO CÓDIGO (POR GRUPO)
- APRESENTAÇÃO FINAL DOS TRABALHOS

**Passo Opcional*